経済学実験ソフト z-Tree の変数概念

飯 田 善 郎

要　　約

経済学実験ソフト z-Tree の変数概念を解説する。z-Tree においては変数がテーブルと言う概念で区別され、テーブルが異なると変数の実行上の振る舞いが違ってくる。テーブルの機能とテーブル間のアクセスに関するルールについての説明を行う。

キーワード：実験経済学、プログラム

1．初めに

z-Tree（Zurich Toolbox for Readymade Economic Experiments）は、Zurich 大学の Urs Fischbacher によって開発された実験経済学用ソフトウェアである。対話型のインターフェースによる実験プログラムが可能で、その汎用性の高さとプログラミングの容易さから広く世界中の実験経済学者に利用されている。

z-Tree は基本的なプログラミングにおいては極めてハードルが低く、直感的な記述で目的を達する事ができる。z-Tree の変数概念上の特徴は、変数を定義する際その機能的属性も同時に定義することである。(機能的属性の定義は変数をなんという種類のテーブルに置くかを指定することで行う。)この機能が実験に適したものであるため、機能的属性の定義はプログラミングにおける実験者の労力を劇的に引き下げる。それは特に単純なプログラムでは有意義である。しかし機能的属性の異なる変数間や異なる時点間でのデータの受け渡しが必要なやや複雑なプログラムをする場合、機能的属性の相互関係を把握していなければならない。z-Tree 上のプログラミングにおける躓きは多くの場合ここから生じると考えられる。z-Tree のチュートリアルマニュアルは Web 上で公開されているが[1)]、その説明は属性間の相互関係や属性そのものを把握させるには十分とは言いがたい。本稿はプログラミング全般を解説するものではなく、変数概念のみに焦点を絞って解説し、この問題を補うものである。

２．テーブルの概念と変数の属性

（1）テーブル

z-Tree でのプログラミングは、ステージツリーと呼ばれる流れ図を画面上で組み立てる形で行う。流れ図の構成要素はエレメントと呼ばれる。変数を扱う場合、プログラムエレメントと呼ばれるエレメントを適切な箇所に挿入し、そのダイアログボックス内に割り当て構文（ステートメント）を記述する形で行う。このダイヤログボックスにテーブルを指示するプルダウンメニューがあり、実験者は必ずどのテーブル上で変数を扱うのかを決めなくてはならない。（図１）このテーブルの指定がそのまま変数の属性の指定となる。囚人のジレンマや公共財実験など一般的な実験において用いられるのは Globals, Subject、Session、Summary の４つのテーブルである。Summary は後述する事にし、ここではまず一般的なプログラミングでは必ず用いる Subjects テーブルについて説明し、次に Globals テーブルと Session テーブルの関係を示す。

図１　テーブルを決めるプルダウンメニュー

（2）Subjects テーブル

今、試行回数が３回の２人ゲームを被験者２名で行うとする。被験者の得点を Score という変数に入力させる事にするなら、一般的なプログラム言語においては

Score（被験者番号、試行回数）

という配列変数を用意する必要がある。この例では２×３＝６の６つの変数を定義しなければならない。

しかし z-Tree のプログラムダイヤログボックス内で Subjects テーブルを指定して変数の割り当て構文を記述するとき、このような配列変数を用いる必要はない。Score がどう決まるかを定義すれば、それぞれの被験者の試行回数ごとの Score が計算されて記録に残される。これは Subjects テーブルが次のような形で変数を扱うからである。

まず試行回数を書く必要がないのは、変数は常に現在試行されているピリオドの変数とみなされるからである。z-Tree では試行を何回も繰り返す場合、バックグラウンドと呼ばれるアイコン内のダイヤログボックスに試行回数を入力する事でそれを実現する。ユーザーは繰り返し実験でも１回分のプログラムだけを組み、繰り返しの回数はダイヤログボックスで指定する。通常のプログラム言語であればユーザーが試行回数を引数に持つ配列変数を用意し、一回終わるたびに引数を書き換えるプログラムを書く必要があるが、z-Tree のこの仕様はこうした手間を取り除く。

z-Tree が試行回数を管理するため、ユーザーがダイヤログボックス内で扱う変数は常にカレントピリオドと呼ばれる、z-Tree が現在実行中ピリオド限りの変数であるとみなされる。このため同じ変数 Score でも１回目の試行の Score と２回目の試行の Score は別のものとみなされ、それぞれに計算され、記録される。

この仕様のおかげで回数を管理するプログラムを書く必要がなくなる反面、変数は常に現在の試行に於ける変数とみなされてしまうため、プログラム上で過去のデータにアクセスできなくなると言う問題が生じる。過去のデータにアクセスするためには専用の関数を用いるか、Session テーブルを使う必要があるが、それについては後述する。

被験者番号を書く必要がないのは、Subjects テーブル上に変数を定義すると、プログラム上の記述では同じ変数でも実行上は被験者ごとに異なる変数であると扱われるからである。Subjects テーブルは被験者ひとりひとりに割り当てられるレコードと呼ばれる小単位で構成される。変数は常にその瞬間 z-Tree が計算しているレコード固有の変数とみなされる。同じ変数 Score でも被験者番号１の被験者のレコード上の Score と被験者番号２の被験者のレコード上の Score は別の変数とみなされ、個別に計算される。このため実験者は被験者数に合わせた配列変数を用意し、全ての被験者について計算するようにプログラムする手間がなく、原則的に対称な関係の被験者ならば一人分のプログラムを書くだけでよい。

この仕様はプログラムで個々の被験者を個別に管理する必要がなくなる反面、他のプレイヤの変数が直接参照できないと言う問題を生じさせる。ゲームの結果を出すためには自分（当該プレイヤ）の

意思決定の他に相手プレイヤの意思決定の情報が必要であるが、通常 Subjects テーブルのあるレコード上で見つけられる変数は上記の仕様のために当該プレイヤのレコードの変数だけである。相手プレイヤの意思決定を知るにはテーブル関数と呼ばれる関数を用いる必要がある。これについては後述する。

（3）Globals テーブル

Globals テーブルは、生産関数のパラメータや初期保有など全被験者にとって共通の変数を定義するために用いる。このためグローバルテーブルに含まれるレコードはひとつだけである。Globals テーブル上の変数も Subjects テーブルと同様にカレントレコードのみを見るため、試行回数を被験者が管理する必要はない。

（4）Session テーブル

Subjects テーブルや Globals テーブルで定義された変数は、１回の試行が終了すると（一般的にはステージツリーの先頭から最後までを経過すると）変数は全て記録された上でリセットされる。複数回の試行の間リセットされない変数が必要な場合、それは Session テーブル上に定義することで実現される。Session テーブルは基本的に被験者ごとに定義される。ここで注意すべきなのは Session テーブルに書かれたステートメントそのものは毎試行回ごとに実行されることである。したがって複数のピリオドにわたって保持したい値がある場合次の回のステートメントの実行によってそれが上書きされてしまわないようにしなければならない。例えば実験開始直後にランダムで決めた特定のピリオドのみに被験者の環境が変わるようなトリートメントつくりたいとしよう。環境を変えるピリオドの値を保持するため Session テーブルで乱数発生のステートメントを書くなら、そのステートメントはピリオドが１回目のときのみ実行されるよう条件文を付け加えないと、データを保持するという Session テーブルの機能は有効に機能しない。そうしなかった場合、１回目のピリオドでそのステートメントが実行されて環境変化のピリオドが定義されても、２回目が始まると Session テーブル内の乱数発生のステートメントが再び実行されるため、前回の結果を上書きしてしまうからである。

（5）各テーブルの実験実施における機能

Subjects, Globals, Session の３つのテーブルは上述のように１）いつ変数がリセットされるか２）各被験者が個別のレコードを持つか否か、の２つのベクトルから区別される。

３人の被験者が３ピリオドの実験を行う場合のテーブルの模式図を図２に示す。

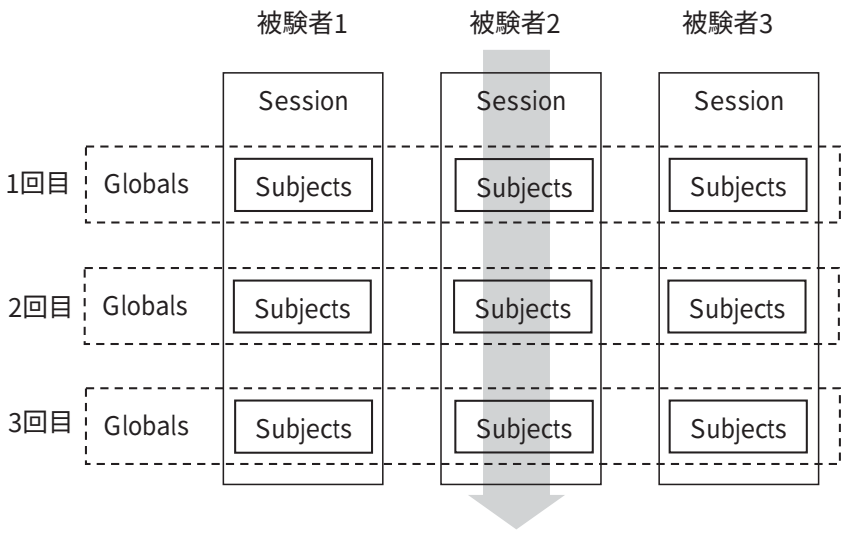

図２　テーブル概念図

この図において箱は点線、実線にかかわらずそれぞれ個別のレコードを表す。レコードは３ピリオドの実験を行うこの例では全部で 15 個あるということになる。Subjects テーブルでは各被験者にレコードが用意されるのでピリオドごとにレコードは３つ用意され、被験者共通である Globals テーブルではレコードはひとつ用意される。ピリオドが変わるとそれらはリセットされ、同じものがまた用意される。Session テーブルは被験者ごとにレコードが用意され、その実験が終了してアンケートが終了するまでリセットされない。

それぞれのレコードは独立しているため、同じ変数名をそれぞれのレコードで使っても別の変数として扱われる。例えば Subjects テーブルで X という変数を定義し、Globals テーブル、Session テーブルでも X を定義するとこの例では 15 個の同名の変数 X が存在することになる。z-Tree はその 15 個を別個の変数として扱い、計算し、結果ファイルにはテーブル名、被験者番号、ピリオド数で区別しながら 15 個分の X の内容を書き出す[2)]。

図では Subjects テーブルのレコードの箱が Globals テーブルや Session テーブルの箱の中に入っているが、これは Subjects テーブルが Globals テーブルや Session テーブルの一部であるという包含関係を意味するのではなく、Subjcets テーブルの各レコードを基準としてみたときの変数の共有を意味している。例えば同一のピリオドにおいて各被験者は Globals テーブル上の同じ変数を共有している。同様にある被験者にとってはセッションが終了するまで Session テーブル上の変数はどのピリオドでアクセスしても（代入された数値は変わっているかもしれないが）同じ変数である。

また、この箱はテーブルのデータがどの時点でリセットされるかも表す。灰色の矢に沿って実験が経過するとしよう。実験が開始され、第１回目の試行に入ると（矢印が Subject, Globals, Session の

箱に入ると）Subject, Globals, Session テーブル上のステートメントが実行され、変数が定義される[3)]。１回目の試行が終わると（矢印が Subject, Globals の箱の外に出ると）Subject, Globals テーブル上の変数はリセットされ、全て０に戻る。Session テーブルの変数は試行ごとにはリセットされず、セッションのあとのアンケートが終了するまで変数の値が保持される。第２回目の試行が始まり、矢印が２行目の箱に入ると再び Subejct, Globals, Session テーブル上のプログラムが実行される。

このように、テーブル上の変数がどの被験者に属し、ステートメントがいつ実行され、いつリセットされるかはどのテーブル上で変数が定義されているかで決まる。これをまとめたのが表１である。

表１　各テーブルの特徴

	Subjects	Globals	Session	Summary
変数データが各被験者に属する	○	×	○	×
ピリオドが終わるたびに変数をリセット	○	○	×	○
ピリオドごとにプログラムを実行	○	○	○	○
過去の履歴を実行中観察できる	×	×	×	○

*Summary テーブルについては後述する。

３．データのアクセス

（１）テーブル・レコードをまたぐデータのアクセス

z-Tree のプログラムにおいては必要な機能に合わせてテーブルを選ぶが、当然テーブル間あるいはレコード間でデータを受け渡しする必要が出てくる。z-Tree では直接他のレコードやテーブルの値を参照できる場合とできない場合があり、注意が必要である。

データのアクセスは、２つのベクトルから整理しておく必要がある。ひとつはテーブルの種類が同じで、異なる被験者同士のレコード間でのデータの受け渡しである。たとえば被験者の固有の意思決定は Subjects テーブルに記述されるが、ある被験者から見て自分の Subjects テーブルに記述されているのは当人の意思決定だけである。自分以外の被験者の Subjects テーブルの内容は別の Subjects テーブルに記述されていて、直接参照ができないため、なんらかの方法でアクセスする必要がある。

もうひとつのベクトルは、同じ被験者が異なる種類のテーブル間でデータを行き来させる場合である。先の例で言えば Session テーブルで定義した特定のピリオドでのみ被験者の環境を変えるような場合、被験者固有の環境は Subjects テーブルに記述されるが、いつ変化させるかを決めるためには、環境を定義する Subjects テーブルから Session テーブルを参照する必要がある。この章ではこれらのテーブルおよびレコード間のデータの受け渡しを可能にする方法について述べる。

（２）同じテーブル内の異なるレコード間でのデータのアクセス

同じ種類で異なる被験者に属するデータの相互アクセスを可能にするのがテーブル関数である。テ

ーブル関数は z-Tree の変数概念が生んだ特有の関数であり、主なものは次の表２にまとめられる。

表２　主なテーブル関数

テーブル関数	関数の機能
sum（条件、変数）	条件を満たす変数の合計
find（条件、変数）	条件を満たす変数の値
count（条件）	条件を満たす被験者の数
maximum（条件、変数）	条件を満たす変数の中で最大の値
minimum（条件、変数）	条件を満たす変数の中で最小の値

テーブル関数の使用例を説明する。例えばある被験者から見て自分と同じグループの被験者の Contribution の合計を SumC に代入したいときには、プログラムダイアログボックスに

SumC=sum (Group==:Group, Contribution);

と記述する。条件文右辺の Group の前のコロン：に注意されたい。これはスコープオペレータとよばれ、「自分の」という意味を変数に付与する。つまり： Group は「自分の Group 番号」を意味する。sum (Group==:Group, Contribution) という関数を与えられると z-Tree は全ての被験者のレコードを見てまわり、「当の被験者のグループ番号と同じグループ番号」という条件を満たすレコードを持つ被験者の Contribution を（当人の分を含めて）全部拾ってきて合計を計算し、SumC に代入する。

もし同じグループで最も公共財への貢献量が少ない人の貢献量を見つけてきて Parsimony という変数に代入したい場合は次のようにする。

Parsimony=minimum (Group==:Group, Contribution);

「自分の変数と同じ変数をもつ」という条件はあまりによく使われるため、same（変数）という関数が作られている。グループの貢献量の合計はこれを用いることで

SumC=sum (same (Group), Contribution);

と言う形で記述される。

（３）異なる種類のテーブル間でのデータのアクセス

次に異なる種類のテーブル間でのデータの受け渡しであるが、これは２つの方法がある。ひとつは直接参照である。たとえば Globals テーブルで定義された変数はそのまま Subjects テーブルや

Session テーブルで使用することができる。しかし Globals テーブルから Subjects テーブルを見る場合、どの被験者の Subjects テーブルを見るか、すなわちどのレコードの変数を見るのかという条件を与えなくてはいけない。ここで再びテーブル関数が登場する。テーブル間でのデータのアクセスのもうひとつの方法は、"テーブル名.テーブル関数"である。たとえばあるピリオドに被験者の環境が変わる実験で、変わるピリオドが Session テーブル上の変数 ShockPeriod に定義されていて、それを Subjects テーブル上の変数 Upset に代入したい場合、Subjects テーブルのプログラムダイヤログボックス内で次のように記述する。

Upset=session.find (same (Subject), ShockPeriod);

このステートメントに行き当たった z-Tree は、Session テーブルにゆき、当該被験者の被験者番号（Subject）と同じ番号のレコードを探しだし、そこに定義された ShockPeriod の値を拾ってきて Upset に代入する。もうひとつ例を挙げよう。Globals テーブル上の変数 RepDec に被験者番号（Subject）１番の被験者の意思決定 Decision を代入したい場合、次のように記述する。

RepDec=subject.find (Subject==1,Decision);

このステートメントでは z-Tree は Subjects テーブルのレコード内の変数 Subject が１である被験者を探し出し、そのレコードにある変数 Decision の値を拾ってきて RepDec に代入する[4)]。

テーブル間のデータのアクセスで、直接参照できる場合と"テーブル名.テーブル関数"が必要になる場合の関係は図３のように整理できる。

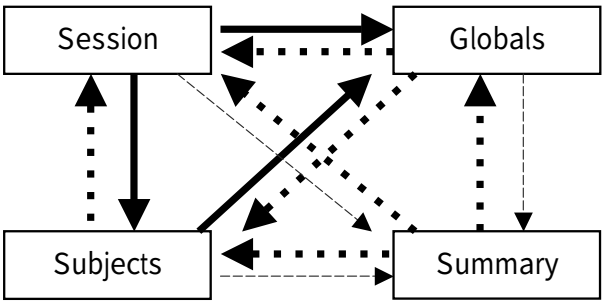

図３　テーブル間のアクセス

* 太い実線は直接アクセスできる関係
* 点線は“ テーブル名．テーブル関数 ”でアクセスできる関係
* 細い点線は“ テーブル名．テーブル関数 ”でアクセスできるが、１回目以降更新されない

異なる種類のテーブルの変数にアクセスするとき、どのような場合に直接参照でき、どの場合に“テーブル名．テーブル関数”が必要かは一概には言えない。しかし、おおよそ直感的にはどの被験者のどの瞬間のデータかを厳密に指定する必要がある場合には“テーブル名．テーブル関数”が必要になるといっていいだろう。たとえば Subjects テーブルから見て Globals テーブルのデータは全員にとって共通で、過去の値はリセットされているので値はひとつしかない。従ってどの変数かを指定する必要はない。しかし Globals テーブルから見て Subjects テーブルに書かれた変数は被験者の数だけある。したがってどの変数なのかを指定する必要がある。

４．異なる時点間のデータのアクセス

既述のように、Session テーブル以外のテーブルはピリオドごとにデータがリセットされてしまうため、当該ピリオドの変数以外は直接アクセスできない。過去のデータにアクセスしたい場合は、必要なデータを Session テーブルに渡しておくか、OLDsubjects を用いることになる。OLDsubjects をテーブル関数の前におく事で、Subjects テーブルの１期前のピリオドにのみアクセスできる．たとえば１期前の自分の Decision を PrevDecision に代入したい場合、次のように記述する。

```
PrevDecision=OLDsubjects.find (same (subject), Decision);
```

これを応用すれば特定の数値をセッションの終了まで保持する事ができる。Session テーブルの機能はこれで代替できるが、Subjects テーブル上のデータはピリオドが終わると強制的にリセットされてしまう。ひとつのセッションの結果を次のセッションに持ち込みたい場合や、アンケートの内容に実験結果を反映させたい場合にはやはりセッションが終わってもデータがリセットされない Session テーブルを使う必要がある。

５．Summary テーブル

Summary テーブルの意義は、主に実験中の被験者行動を時系列に沿って監視できるということにある。（この点マニュアルの記述がややミスリーディングなところがあり、把握しにくい。）z-Tree では実験者は実験中、Globals や Session、Subjects テーブル内のデータを専用のウィンドウで監視できるが、監視できるのは当該ピリオドのみで、実験中に過去のデータを遡って見る事はできない。一方 Summary テーブルのウィンドウ内では過去のピリオドも表示されるため、実験中に時系列に沿って変動を把握したい数値がある場合、その数値を Summary テーブル上の変数に渡すことで、過去のデータを画面上に表示させ続ける事ができる。Summary テーブルはこのように実験実施上の便宜を提供するところにその機能の主旨があり、このテーブルを積極的にプログラムに利用する意義は乏し

い。基本的に Summary テーブルは Subjects テーブルや Session テーブルと異なり、被験者ごとの個別の変数は扱えない。その意味では Globals テーブルに近いが、Globals テーブルと異なり、他のテーブルからは Summary テーブルのデータにアクセスするにはテーブル名とテーブル関数を介さねばならない。

Summary テーブルは被験者ごとに用意されているものではないので、参加者の利得の平均値など、名前どおり全体の概要を記述する使い方が一般に考えられる。しかし実験中に特定の個人の行動を最初から最後までウィンドウ上に表示させることも個人の行動を Summary テーブルに移せば可能である。

６．異なるテーブルで同じ変数を使うことによる問題点

z-Tree では同じ変数名を異なるテーブルで使うことができる。その場合変数名は同じでもそれぞれ別の変数とみなされ、個別に計算と記録が行われるため、プログラムの実行上は変数名の重複は問題とならない。しかしこれは以下の２点から避けるべきである。第１に同じ表記で異なる変数が同じトリートメントプログラム上にあるとプログラムする側の混乱と過ちを招きやすい。第２に、画面上にその数値を表示させる場合、誤った変数の方が表示されてしまうことがありうる。基本的に画面上に表示させる事ができる変数は Subjects テーブルと Globals テーブル上の変数だけであるが、同じ変数名が両方のテーブルにある場合、どちらの変数を表示するかを指示する方法はなく、Subjects テーブルの変数の方が表示されてしまう。従って Globals テーブル上の変数を表示させるつもりが Subjects テーブル上の変数を表示させてしまうことがありうる。

７．Parameters テーブル

（１）Parameters テーブルの概要

基本的に z-Tree の Subjects テーブルと Globals テーブルは、被験者が対称的で、同じ実験を繰り返す場合に便利な設計になっている。しかし被験者が複数のタイプに分かれているような実験、あるいは実験の設定がピリオドによって異なるような実験を行いたい場合もありうる。このような場合に Parameter テーブルが用いられる。

Parametesr テーブルはメニューから別ウィンドウで展開される図４のような表で、図では S1, S2, S3 のラベルがついている列ラベル、1,2,3 のラベルがついている行ラベル、左肩にグループ番号を示す小さな箱かついたセルで構成され、それぞれクリックするとプログラムを書き込めるダイヤログボックスが開く。

列ラベルをクリックして現れるダイヤログボックスは Role Parameters といい、そこに記述したパラメータ設定などのステートメントはその列番号の被験者に対して全期間を通じて実行される。

行ラベルをクリックして現れるダイヤログボックスは Period Parameters といい、そこに記述したステートメントはそのピリオドの全被験者に対して実行される。

セルをクリックして現れるダイヤログボックスは Specific Parameters で、記述されるステートメントはその被験者のそのピリオドでのみ実行される。

Parameters テーブルは応用範囲の広い機能であるが、その名前から Subjects テーブルや Globals テーブルと同様の独立した変数テーブルであるかのように誤解しやすい。実際はこの３つのパラメータは既存のテーブルに属するものである。どのテーブルに属しているかで３つのパラメータの機能と相互関係が決まってくる。以下この点を説明してゆく。

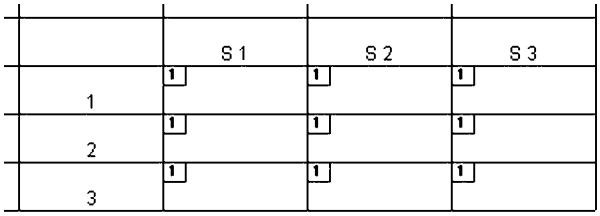

図４　Parameters テーブルウィンドウ

（２）Role Parameters と Specific Parameters

Role Parameters、Specific Parameters 内の変数は、Subjects テーブル内の変数として扱われる。実行においては Specific Parameters の内容よりも Role Parameters の内容の方が優先される。従って Role Parameters で定義したある数値を特定のピリオドだけ Specific Parameters を使って変更する事はできない。

Role Parameters、Specific Parameters は Subjects テーブル内の変数なので、他のテーブルにアクセスする手段は３節で説明した Subjects テーブルのそれと同じである。すなわち Globals テーブル、当該被験者の Subjects テーブルの変数には直接アクセスできるが、自分以外の被験者のレコードにはテーブル関数が必要であり、Session テーブルの変数には“ テーブル名．テーブル関数 ”が必要である。Globals テーブルと Subjects テーブルに同じ変数がある場合、Role Parameters、Specific Parameters からその変数を参照すると Subjects テーブルの変数の値が返される。これはこの２つのパラメータが Subjects テーブルに属することから当然の帰結である。Globals テーブルの変数を Subjects テーブルよりも優先して参照させるためには、“ テーブル名．テーブル関数 ”を用いる必要がある。

（３）Period Parameters

Period Parameters 内の変数は Globals テーブルの変数として扱われる。Period Parameters で全員

共通で定義した数値のうち、特定の被験者の分だけを Specific Parameters で変える事はできないので注意が必要である。これは Period Parameters 内で定義した数値は Globals テーブル上の数値として扱われ、Specific Parameters 内で定義した数値は Subjects テーブル上の数値として扱われるからである。テーブルが異なる事から同じ変数を定義しても別個の変数として計算されるのである。

Peiod Parameters のダイヤログボックスからは Globals テーブルには直接アクセスできるが、Subjects テーブルと Session テーブルにはテーブル名とテーブル関数を挟んでアクセスする必要がある。これらの特徴は表３にまとめられる。

表３　Parameter テーブルの機能

パラメータ名	Role	Period	Specific
定義する場所	列ラベル	行ラベル	セル
効果範囲	その列の被験者に対して全ピリオド	そのピリオドの全被験者	その列のそのピリオドの被験者
テーブル	Subject	Globals	Subject
備考	あらかじめステージツリーで定義した変数のみ扱える	Role、Specific とはテーブルが違うので同じ変数名を使っても違う変数とみなされる	Role、Period で定義されている事を Specific で定義しなおすことはできない

８．終わりに

z-Tree のプログラムの変数概念について詳述してきた。残念ながら公開されているチュートリアルマニュアルにおいてはこの概念の説明は十分とはいえない。変数がテーブルと言う概念のもとで扱われる事、テーブルごとに属性が異なる事、テーブルが異なれば同じ変数名でも異なる変数と扱われる事などは特に一般的なプログラム言語から見ると特異なものと考えられる。しかしこの点を把握する事で z-Tree でのプログラム上の困難はほぼ排する事ができると思われる。

なお、本稿においては市場実験で用いられる Contracts テーブルについては解説しなかった。これは今回説明した４つのテーブルとはやや独立した位置づけになり、関連性が弱い事やチュートリアルマニュアルの説明でほぼ問題がないと思われるからであるが、もし必要があれば機会を改めて詳述したい。

注

１）http://www.iew.unizh.ch/ztree/support.php#mailinglist

２）無論、これは極端な例であり、後述のように異なる種類のテーブルで同じ変数を使うことは薦められない。

３）実際の各テーブルのプログラムの実行順序は、ステージツリー上にどの順番でプログラムアイコンを並べられているかに依存する。

4）このようにレコード間でデータを行き来させる場合、レコードを指示するのに使われるのが Subject や Group という変数である。これらの変数は実験者が与えなくても自動的に z-Tree が各被験者（各レコード）に賦与する。実験者がこの変数の値を操作することも可能で、Subject 変数はメニューから Client's table を呼び出すことで、Group 変数はプログラムで適当なグループ変数を与える事で変更できる。

Variable Concepts of z-Tree

Yoshio IIDA

Abstract

This paper provides detailed information about variable concepts of z-Tree, one of the most popular economic experiment software. In z-Tree, each variable belongs to one of the Tables that regulate function of the variable during the experiment. The functions of the Tables and rules of accessibility between the Tables are explained.

Keywords : Experimental Economics, Program